Præsentation af FURPS+ krav til applikationen ”artists-CRUD-app”

- **Functionality:** Applikationen skal opfylde kravene for CRUD-operationer, filtrering, sortering og præsentation af kunstnerdata. Funktionaliteten skal være pålidelig og intuitiv.

Generelt i functionality:

CRUD: Der er Create, Read, Update og Delete i webapplikationen og de er alle nemme at tilgå både front- og backend.

Filtrering: Det er nemt og hurtigt at navigere i sortering- samt filtreringsværktøjet, som man kan finde øverst på websiden. Hvis man klikker på filtreringsknappen, kan man filtrere med udvalgte musikgenre. Derudover, hvis man klikker på ”Favorite”-knappen kan man se de kunstnere, som man har favoriseret, og man kan fjerne dem igen hvis dette ønskes.

Sortering: Der kan sorteres nemt ved at klikke på ”Sort By”, hvori der kommer en drop-down menu med de valgmuligheder der er sat op. Disse er også efter alfabetisk rækkefølge.

Præsentation af kunstner data: Disse kan man få vist som navne i en liste hvis man klikker på den pågældende kunstners billede. Der åbnes et modal vindue op hvor kunstnerens informationer står beskrevet.

Der er blevet udover dette også tilføjet en funktion i højre hjørne på hjemmesiden som der gør at man kan søge på alle artisterne.

Frontend:

Brugergrænsefladen er brugt ved hjælp af HTML, CSS og JavaScript. Derudover er koden opdelt i modules.

Backend:

Der er blevet implementeret et REST API når man bruger Node.js og Express.js
http-metoderne GET, POST, (PUT/PATCH) og DELETE har fået ruter for tilhørende endpoints.

- **Usability:** Brugergrænsefladen skal være brugervenlig og nem at navigere. Alle CRUD-operationer skal være let tilgængelige, og filtrerings/sorteringsfunktionerne skal være intuitive

Generelt i usability:

Min brugergrænseflade er nem at navigere i, da der er overblik lige fra starten når man kører appen. Sortering-, filtrering- og favoritfunktionerne er alle nemme at bruge og forstå, da disse valgmuligheder står på samme sted i applikationen. Det foregår intuitivt når man bruger menuen.

Frontend:

Rent brugervenlighedsmæssigt er der gjort fokus på en nem navigation på brugergrænsefladen.

CRUD-operationer bliver præsenteret på en nem og ordentlig måde som er intuitivt for brugeren.

- **Reliability:** Applikationen skal være stabil og pålidelig. Dataintegritet og korrekt håndtering af CRUD-operationer er afgørende.

Generelt:

Appen starter hurtigt op og giver fejlbeskeder tilbage til modtageren når den skal gøre det og min API er specifik i hvad den godtager i dens operationer.

Frontend:

Der er en blevet foretaget en grundig test for en fejlfri oplevelse og taget højde for funktionalitetsfejl.

Backend:

De forskellige scenarier som kan forekomme er blevet testet i forhold til pålidelighed.

- **Performance:** Applikationen skal have an acceptabel ydeevne, herunder hurtig datahentning og responsivitet i brugergrænsefladen.

Generelt:

Webapplikationen er hurtig når man sletter, læser, klikker, opretter og opdaterer kunstnerne. Dens API kald fungerer som de skal. API kaldene kommer hurtigt så applikationen kører godt og det tager omkring 10 millisekunder før der eksekveres en handling.

Frontend:

Der forekommer en hurtig indhentning af data.

Backend:

For at tage udgangspunkt i HTTP-anmodninger og datahåndtering er backenden blevet optimeret for at sikre en tilfredsstillende ydeevne.

- **Supportability:** Koden skal være velstruktureret og veldokumenteret, så det er nemt for fremtidige udviklere at vedligeholde og udvide applikationen.

Generelt:

Der er opsat kommentarer op i koden for at gøre det nem for udviklerne at rette i min kode, og funktionerne har navne, som er med til at hjælpe den som læser min kode med at forstå hvad koden gør. Kommentarerne er tilføjet de steder hvor det er relevant for den person som måske kigger på kode i fremtiden, så han/hun nemt tilgår koden via frontend / backend.

- **+:** Der skal også tages hensyn til andre relevante krav og overvejelser for at sikre en vellykket udvikling og implementering af applikationen. Ud over FURPS-kravene skal udviklingen også overholde generelle principper for kodestruktur, modularitet og afhængighedsstyring, herunder Separation of Concerns, Loose Coupling og High Cohesion.

Funktionerne er bygget op på den måde at de fleste er bundet sammen med et API. Det betyder at funktionerne skulle ændres hvis der bliver tilgået et API kald. Funktionerne er så uafhængige af hinanden som muligt. Videreudvikling af applikationen er også nemt at tilgå.

Der er god CRUD kodestruktur og funktionerne er generelt opdelt på en måde hvorpå det er nemt at finde rundt i modules. Der er oprettet flere modules, som hver har deres eget formål. Tingene er adskilt så meget som muligt, og applikationen bruger forskellige funktioner og moduler til at manipulere og vise data.

I forhold til Loose Coupling er funktionerne som tidligere nævnt så uafhængige af hinanden som muligt. Funktionerne der arbejder sammen er sat tæt på hinanden i hht. High Cohesion.